ISD Unit Test Guideline

Number: 580-GL-062-01
Effective Date: August 28, 2006
Expiration Date: August 28, 2011

Approved By: (signature)
Name: Barbara Pfarr
Title: Assoc. Chief, ISD

Responsible Office: 580/Information Systems Division (ISD)
Title: ISD Unit Test Guideline

Asset Type: Guideline
PAL Number: 2.4.2.2

Purpose

The purpose of this guideline is to provide advice and suggestions for the conduct of unit testing within the Information Systems Division (ISD).

Scope

This guideline is intended for use on all ISD software projects, or all Mission projects for which ISD is providing software support. This guideline is applicable to unit testing of both newly developed units and units that have been modified during development or maintenance.

GUIDANCE: Note that, on some projects, a "unit test," or "unit test framework," is a software tool that calls the various functions in the unit being tested. The tool automatically tests the paths in the unit, compares the test results to the expected results, and highlights any discrepancies encountered. While such tools may be quite useful, they're beyond the scope of this Guideline. This Guideline addresses the situation where no such tool is available and human insight is needed to plan and conduct the unit testing.

Guideline

The intent of unit testing is to confirm that a unit performs the capability assigned to it, correctly interfaces with other units and data, and represents a faithful implementation of the unit design. Each software unit should be tested before being integrated with other units. Unit testing should be planned in advance, and is generally documented in a unit test plan.

This guideline uses the following terms and definitions:

Unit test plan: A detailed description of the tests to be performed on a software unit to verify its function and logic paths.

Unit test case: A set of test inputs and expected results developed to verify that a specific function or logical path of a software unit performs in accordance with its requirements.

For the purpose of this guideline, a **unit** is defined as:

(1) A separately testable element specified in the design of a computer software component. (2) A logically separable part of a computer program. (3) A software component that is not subdivided into other components. (From IEEE Std 610.12-1990)

The following steps are recommended during unit testing:

- Plan the unit tests. The unit test plan:
  - Identifies the unit(s) to be tested
  - Indicates the unit testing approach (e.g., functional, path, or statistical testing) that will be used to develop and carry out the plan
  - Specifies the unit test execution environment (e.g., test platform, stubs and/or drivers to be used, any test support or measurement software required)
  - Defines the unit test cases to be run (i.e., the function or path to be tested, the input to the unit test case, and the expected results)
  - Generally provides a checklist or other vehicle for a peer of the developer to use when reviewing the unit test results.
- Tailor the unit test plan for the type of unit being tested. There are three basic types of testing:
  - Structural, or path, testing, used primarily for units that affect the flow of control through the software system. The unit test plan should include enough unit test cases that each logic path in the unit design is executed at least once.
  - Analytical testing may be more appropriate for a unit that performs complex mathematical, physical, or astronomical computations.
  - User interface testing thoroughly exercises user inputs.

  Unit test cases should verify that each logical path of the code meets all requirements for that logic. They should verify boundary conditions and error situations as well as nominal cases.
- Perform the unit tests; this is generally done by the developer, but for very large systems, or systems subject to stringent safety or security requirements, may be performed by independent unit testers.
- If errors in code are detected, the developer corrects them and the unit is retested. Errors that trace to design or requirements must be handled by re-applying the appropriate design or requirements process.
- Review the unit test results. This is generally done by a peer of the developer, or perhaps the developer's team leader.
  - This review involves checking the unit test results against the unit test plan to ensure that all logic paths, computations, or user options have been exercised
  - The review also includes verifying that the unit test results are accurate and correct.
  - If errors are uncovered in the review, the unit is returned to the developer for correction and retesting.
  - Otherwise, the reviewer certifies that testing of the unit is complete.

- The unit test plan and unit test results are stored in the Software Development Folder for the unit. The Software Development Folder is frequently maintained online.
- If extensive changes are made to the unit at a later time, the unit code must be reinspected, and the unit must then be retested and recertified.

Some projects may choose to substitute rigorous inspections for unit testing. For systems with the highest cost of failure (e.g., flight software), both rigorous inspections and unit tests are recommended.

Measures

Recommended Measures:

- Number of units tested and certified, as a fraction or percentage of number of units coded
- Staff-hours devoted to unit testing and certification

Required Measures: None

References

- **Glossary:** http://software.gsfc.nasa.gov/glossary.cfm
  Defines common terms used in ISD processes
- **Process Asset Library:** http://software.gsfc.nasa.gov/process.cfm
  Library of all ISD process descriptions
- FSW Unit Test Standard, Flight Software Branch – Code 582, 582-2000-002, PAL #2.4.2.2.1.
- Recommended Approach to Software Development, Revision 3, SEL-81-305, Goddard Space Flight Center, June 1992.

Change History

Version	Date	Description of Improvements
1.0	8/23/06	Initial approved version by CCB